A Bayesian Approach to Parameter Inference in Queueing Networks

A Bayesian Approach to Parameter Inference in Queueing Networks

Weikun Wang, Imperial College London, UK, weikun.wang11@imperial.ac.uk
Giuliano Casale, Imperial College London, UK, g.casale@imperial.ac.uk
Charles Sutton, University of Edinburgh, UK, csutton@inf.ed.ac.uk

The application of queueing network models to real-world applications often involves the task of estimating the service demand placed by requests at queueing nodes. In this paper, we propose a methodology to estimate service demands in closed multi-class queueing networks based on Gibbs sampling. Our methodology requires measurements of the number of jobs at resources and can accept prior probabilities on the demands.

Gibbs sampling is challenging to apply to estimation problems for queueing networks since it requires to efficiently evaluate a likelihood function on the measured data. This likelihood function depends on the equilibrium solution of the network, which is difficult to compute in closed models due to the presence of the normalizing constant of the equilibrium state probabilities. To tackle this obstacle, we define a novel iterative approximation of the normalizing constant and show the improved accuracy of this approach, compared to existing methods, for use in conjunction with Gibbs sampling. We also demonstrate that, as a demand estimation tool, Gibbs sampling outperforms other popular Markov Chain Monte Carlo approximations. Experimental validation based on traces from a cloud application demonstrates the effectiveness of Gibbs sampling for service demand estimation in real-world studies.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]: Modeling Techniques

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Queueing, demand estimation, Gibbs sampling, normalizing constant approximation

1. INTRODUCTION

Queueing networks are popular models for the description of complex stochastic systems, including for example call centers [Koole et al. 2002] and web applications [Urgaonkar et al. 2005]. In real-world studies, the analyst is often left with the problem of assigning realistic values for the parameters of the model, in particular for the *service demand* placed by requests at queueing nodes. The service demand is the cumulative execution time a request seizes from a server, *excluding* contention overheads due to other concurrently executing requests. Unfortunately, real-world measurements often include these overheads, therefore requiring inference to determine the true demand value. In this work, we propose a new methodology to support the inference of service demands. This is a challenging task because it is difficult to filter out contention overheads from measurements, particularly in those cases where monitoring data is aggregated or incomplete.

We here focus on *closed* queueing network models, which are a popular class of stochastic networks. They are popular for example in software system modelling since complex applications are layered and the interactions between layers typically happen under admission control or finite threading limits [Rolia and Sevcik 1995]. For these models, the service demand at each physical resource needs to be determined from empirical data. Existing approaches perform demand inference in different ways: regression-based estimation procedures [Rolia and Vetland 1995], nonlinear numerical optimization based on queueing formulas [Liu et al. 2005], and maximum likelihood methods [Kraft et al. 2009]. Recent comparative studies indicate that existing methods work in a number of cases, but they are not always accurate and there is no clear winner [Kraft et al. 2009; Spinner et al. 2015]. In this paper, we push the boundary of demand estimation methods by presenting a new inference procedure based on Gibbs sampling that applies to multi-class networks. Compared to previous work, our Bayesian methodology requires independent samples of the number and class of jobs

at the resources observed at random times. Compared to utilization data, such queue-length samples can also be obtained when the monitoring cannot fully access physical resources, since they only require to count the number of queued requests in a system or software component. This quantity is also simpler to obtain than response times, which require tracking of individual requests. Unfortunately, there is a shortage of estimation methods that rely only on queue-length data and this paper proposes a technique to fill this gap. Furthermore, by allowing the specification of prior probabilities, our method enables the use of known information about the demands in the estimation procedure, an important feature missing in existing estimation methods. For example, this can be used to exclude infeasible values, such as incorrect zero demand estimates that are frequently returned by linear regression methods [Casale et al. 2008].

Bayesian inference of queueing model parameters from response time measurements has received some attention in the recent literature [Kraft et al. 2009; Sutton and Jordan 2011]. Maximum likelihood methods have shown potential in [Kraft et al. 2009]. However, with the exception of [Sutton and Jordan 2011], classic Bayesian methods such as Markov-Chain Monte Carlo (MCMC) have not been applied before to the problem of parameter estimation in queueing networks. The proposed MCMC method is promising, but compared to the present work it applies to open queueing networks and single class systems.

We define inference based on queue-length data by looking at the equilibrium state distribution of the queueing network model. Differently from response time and utilization distributions, the equilibrium distribution of queue-lengths is easy to deal with since, under the BCMP theorem assumptions, it takes a convenient product-form expression [Bolch et al. 2006]. This tractability makes the BCMP equilibrium distribution a suitable candidate to define a Bayesian estimation method. In order to *efficiently* apply Gibbs sampling to likelihood functions computed with the BCMP product-form, we show that one needs to obtain an approximation of the normalizing constant of the state probabilities. Unfortunately, no efficient approximation exists for the normalizing constant of closed multi-class queueing networks unless one looks at asymptotic limits with many jobs and many queues [Knessl and Tier 1992], but existing limits in this regime cannot express think times (i.e., cumulative delays at $-/GI/\infty$ nodes).

We show that the problem of approximating the normalizing constant can be efficiently solved by applying existing local iterative approximations, such as the Bard-Schweitzer algorithm [Bolch et al. 2006] or AQL [Bolch et al. 2006], to a first-order Taylor expansion of the normalizing constant. We call this method the Taylor Expansion (TE) method for computing the normalizing constant. The TE method provides, as a by-product, a general-purpose approximation algorithm for computing the normalizing constant not based on asymptotic limits. The scope of this approximation algorithm goes outside the span of demand inference, as it allows the approximate solution of arbitrary closed queueing networks. In addition to TE, we introduce a variant of Monte-Carlo integration (IMCI) for the normalizing constant [Ross et al. 1994] based on approximate mean-value analysis (AMVA). We compare TE with this new implementation (IMCI) together with the original MCI proposed by [Ross et al. 1994] and show that TE is more effective for use in Gibbs sampling. Using randomly generated instances and a real-world case study based on a cloud application, we show that our Gibbs demand estimation method is helpful in practical estimation problems. This paper is an extension of the work in [Wang et al. 2013] with a revisited implementation of the Monte Carlo sampling based method for estimating the normalizing constant and new validations against other sampling algorithms.

Summarizing, the main contributions of this work are:

— A novel demand estimation approach that use observations on the state of the system, described by the number of jobs observed at the resources.
— The ability to integrate in the estimation procedure prior information on the service demands.
— A novel deterministic method for approximating the normalizing constant of multi-class product-form queueing networks, which is particularly suited for application in conjunction with Gibbs sampling, but can also be used as a stand-alone algorithm.
— A variant of the Monte-Carlo integration method proposed in [Ross et al. 1994], which uses importance sampling to estimate the normalizing constant. Our variant leverages AMVA methods in the choice of the parameters of importance sampling.
— An experimental study showing that the proposed method makes accurate estimates of normalizing constants and service demands in real-world web applications.

The rest of this paper is organized as follows. Section 2 gives background of the demand estimation problem. In Section 3, we introduce the novel approximation algorithm for the normalizing constant of multi-class queueing networks. In Section 4, a demand estimation based on Gibbs sampling algorithm is proposed. Section 5 and Section 6 evaluate the proposed algorithm against simulation data and a real-world case study, respectively. Related work is discussed in Section 7 and Section 8 concludes the paper.

2. BACKGROUND AND PROBLEM STATEMENT

We consider systems that can be modelled as closed queueing networks, for example web applications [Urgaonkar et al. 2005],[Zhang et al. 2007]. The reference model has M queueing nodes and R classes of jobs. We assume the model to include a population of K_j users for class $j = 1, \ldots, R$ having a known average think time $Z_j > 0$. We assume that service times and scheduling policies follow the BCMP theorem assumptions [Baskett et al. 1975]. This means that the distribution of think times is general i.i.d. and queues can either be processor sharing with general i.i.d. service times or first-come first-served with exponential service distributions, having identical means across classes. We point to [Bolch et al. 2006] for a description of other cases where the BCMP theorem holds. The system is assumed to be at steady-state and samples of its state are assumed to be sufficiently spaced in time to be approximately independent of each other. Moreover, in our experiment we notice that even with small sampling interval the samples can still be treated as approximately independent.

Let n_{ij} be the number of jobs of class j at station i. Define θ_{ij} to be the demand of class j requests at resource i, which is the product of the mean service time of a request (i.e., its execution time when running alone) and the mean number of visits of the request at resource i before completion. Also, let the service demand matrix be $\boldsymbol{\theta} = (\theta_{11}, \ldots, \theta_{MR})$. Under the above assumptions, the probability for state $\boldsymbol{n} = (n_{01}, \ldots, n_{MR})$ admits the following product-form expression [Bolch et al. 2006]

$$P(\boldsymbol{n}|\boldsymbol{\theta}) = \frac{Z_1^{n_{01}}}{n_{01}!} \cdots \frac{Z_R^{n_{0R}}}{n_{0R}!} \prod_{i=1}^{M} n_i! \prod_{j=1}^{R} \frac{\theta_{ij}^{n_{ij}}}{n_{ij}! G(\boldsymbol{\theta})}, \tag{1}$$

where Z_j is the think time of class-j requests at the delay server and $G(\boldsymbol{\theta})$ is the normalizing constant of the state probabilities ensuring that $\sum_{\boldsymbol{n}} P(\boldsymbol{n}|\boldsymbol{\theta}) = 1$. Notice that $G(\boldsymbol{\theta})$ is expressed as a function of the demands $\boldsymbol{\theta}$, as opposed to the more common dependence on the job populations.

The problem under study is the estimation of the θ_{ij} parameters given observations of the state probability (1) and assuming to know the vector of mean think times $\boldsymbol{Z} = (Z_1, \ldots, Z_R)$. The knowledge of the think times is required for the problem to be well-defined, since two closed models having respectively demands θ_{ij} and θ_{ij}/c_j, $1 \leq i \leq$

M, $1 \leq j \leq R$, have identical stationary queue-length distribution for any $c_j > 0$, thus any estimation problem formulated only on the stationary distribution of these models would not be able to distinguish them. In the presence of positive think times $Z_j > 0$, this ambiguity does not arise and we can therefore define an estimator for the unknown demands θ_{ij}.

In a Bayesian setting, (1) may be seen as the likelihood of observing state $\boldsymbol{n}$ given a particular guess of the unknown demands $\boldsymbol{\theta}$. However, the efficient evaluation of (1) is complicated by the presence of the normalizing constant $G(\boldsymbol{\theta})$, which requires expensive algorithms for its computation. Efficiently computing normalizing constants may be performed by several recursive algorithms, e.g., [Bolch et al. 2006; Casale 2006]. However, none of these existing methods can compute $G(\boldsymbol{\theta})$ in polynomial time as the number of jobs, classes and queues grow large simultaneously. Summarizing, in order to tackle the estimation of the service demands θ_{ij} we need to define:

— A method to efficiently evaluate the likelihood function $P(\boldsymbol{n}|\boldsymbol{\theta})$ for different guesses of the unknown demands $\boldsymbol{\theta}$. We address this problem in Section 3 by defining the TE approximation scheme for the normalizing constant $G(\boldsymbol{\theta})$.
— A search strategy for the optimal demand estimate $\boldsymbol{\theta}$, which we address in Section 4 by defining a Gibbs sampling algorithm for demand estimation in closed networks.

3. NORMALIZING CONSTANT APPROXIMATION

3.1. Motivating Problem

Gibbs sampling is a Markov Chain Monte Carlo [Brooks et al. 2011] method, a popular class of algorithms to draw S samples from high dimensional spaces to solve integration problems. In our setting, Gibbs sampling involves the generation of samples of the service demands, whose empirical distribution will provably converge to the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{n})$ of the demands $\boldsymbol{\theta}$. This posterior distribution accounts for both the data and the prior probabilities. In the special case where the prior probabilities on the demands are uniform, the resulting estimator is a maximum likelihood one and therefore the estimation focuses on iteratively evaluating the likelihood function (1). Unfortunately computing the normalizing constant $G(\boldsymbol{\theta})$ in the likelihood (1) can be computationally expensive. This is because, as the number of classes, queues and jobs increases, the direct computation of the equilibrium probability distribution (1) requires an exponentially increasing effort in both time and space [Bolch et al. 2006]. For example, generating just 50 samples for each demand θ_{ij} by Gibbs sampling for a small model with $M = 4$ nodes and $R = 4$ classes can require a time between half hour and one hour on a commodity laptop, mainly due to the cost of computing the normalizing constant by the convolution algorithm [Bolch et al. 2006]. On larger models, it becomes even impossible to compute it for a single likelihood evaluation. Therefore, in order to make Bayesian estimation viable for closed queueing networks, we need techniques to efficiently approximate the value of the normalizing constant.

In the next subsections we compare two strategies to address this issue, a new implementation of an existing approach based on Monte Carlo Integration (MCI) proposed in [Ross et al. 1994] and our novel method based on Taylor expansion (TE).

3.2. Improved Monte Carlo integration (IMCI)

In [Ross et al. 1994], the authors propose a method based on Monte Carlo integration to approximate the normalizing constant. The idea is to express the normalizing constant as a multidimensional integral and approximate it by averaging random samples. In order to reduce the variance of the result, an importance sampling technique based on exponentially distributed samples is defined [Asmussen et al. 2007] by which the variance of the expectation can be significantly reduced. The accuracy of the procedure

grows with the number of samples, which we denote by I_{MCI}, generated by the importance sampling method. Two alternative variance reduction techniques based on antithetic variates and quasi-Monte Carlo methods can be found in [Tuffin 1997].

In detail, the authors in [Ross et al. 1994] exploit the fact that the normalizing constant of a product form queueing network can be expressed as the following multi-dimensional integral

$$G(\boldsymbol{\theta}) = \frac{1}{\prod_{j=1}^{R} K_j!} \int_{\boldsymbol{Q}^+} \exp(-\mathbf{1}\boldsymbol{u}) \prod_{j=1}^{R} (Z_j + \boldsymbol{\theta_j}\boldsymbol{u})^{N_j} \mathrm{d}\boldsymbol{u} \tag{2}$$

where $u = (u_1, \ldots, u_M)'$ and $\boldsymbol{Q}^+ = \{\boldsymbol{u} \in R^M : u_i \geq 0\}$. This integral can be then approximated with Monte Carlo integration and importance sampling by averaging a set of samples, with cardinality I_{MCI}, of the integrand, i.e.,

$$G(\boldsymbol{\theta}) \approx \frac{1}{I_{MCI}} \sum_{s=1}^{I_{MCI}} \frac{\exp(-\mathbf{1}\boldsymbol{V}^s) \prod_{j=1}^{R} (Z_j + \boldsymbol{\theta_j}\boldsymbol{V}^s)^{N_j}}{p(\boldsymbol{V}^s)} \frac{1}{\prod_{j=1}^{R} K_j!} = \frac{1}{I_{MCI}} \sum_{s=1}^{I_{MCI}} H^s = \overline{H} \tag{3}$$

where $\boldsymbol{V}^s = (V_1^s, \ldots, V_M^s)'$ is the s-th sample vector generated from a new proposal distribution $p(\cdot)$ defined over Q^+. The proposal distribution is introduced to reduce the variance on the result by generating samples that are localized in a region of the state space that most contributes to $G(\boldsymbol{\theta})$. Since the value of the normalizing constant can easily exceed the floating-point range, the above expression may be equivalently computed in terms of its logarithm as

$$\log H^s = -\sum_{i=1}^{M} V_i^s + \sum_{j=1}^{R} N_j \log\left(Z_j + \sum_{i=1}^{M} \theta_{ij} V_i^s\right) - \log(p(\boldsymbol{V}^s)) - \sum_{j=1}^{R} \sum_{u=1}^{N_j} \log(u). \tag{4}$$

In [Ross et al. 1994], the authors chose $p(\boldsymbol{V}) = \prod_{i=1}^{M} p_i(V_i)$ and each $p_i(V_i)$ is an exponential probability density function with mean λ_i^{-1}. The authors assume to know the utilization U_i of server i and define $\lambda_i = 1 - U_i$ for near-optimal reduction in variance of sampling $G(\boldsymbol{\theta})$ and U_i is approximated as $\sum_{j=1}^{R} N_j \theta_{ij} (\sum_{t=1}^{M} \theta_{tj})^{-1}$ for each station i. The authors suggest in follow-up work [Ross et al. 1997] to use a pilot run for determining U_i if the system is near critical usage. In particular, λ_i is redefined as

$$\lambda_i = \begin{cases} 1 - U_i, & U_i < 0.9, \\ \dfrac{1}{\sqrt{\max(N_1, \ldots, N_R)}}, & U_i \geq 0.9. \end{cases} \tag{5}$$

However, not full detail is given on how to recompute the utilization in the following steps and the software is no longer available. Hence we recompute the utilization with

$$U_i = \sum_{r=1}^{R} \frac{G(\boldsymbol{N} - \mathbf{1}_r)}{G(\boldsymbol{N})} \theta_{ir} \tag{6}$$

in MCI where $G(\boldsymbol{N} - \mathbf{1}_r)$ is the normalizing constant in a new model created by removing one job of class r from the original model.

However, we have found that it is much more effective, in general, to approximate U_i by the Bard-Schweitzer AMVA algorithm [Bolch et al. 2006]. The Bard-Schweitzer AMVA method approximates the standard MVA by estimating the mean queue length of one job less population $\bar{n}_{ij}(\boldsymbol{K} - \mathbf{1}_r)$ by interpolating $\bar{n}_{ij}(\boldsymbol{K})$ with $(K_r - 1)/K_r$ so as to reduce the total iterations of MVA. Besides this, our MCI implementation also introduces several other improvements compared to the original description in [Ross et al. 1994], as shown in the pseudo code in Algorithm 1. First, it copes with floating-

Algorithm 1 Improved Monte Carlo Integration (IMCI)

Require: I_{MCI}, I_{iter},$\boldsymbol{\theta}$, $\boldsymbol{Z}$, $\boldsymbol{N}$, ρ
  $[\epsilon_{\min}, \epsilon_{\max}]$:= floating-point range bounds
  for $t = 1 : \lfloor I_{MCI}/I_{iter} \rfloor$ **do**
    for $s = 1 : I_{iter}$ **do**
      estimate utilization U_i with the Bard-Schweitzer algorithm for each station i
      generate V_i^s from an exponential distribution parametrized with $(1 - U_i)^{-1}$
      generate $\log H^s$ by (4) using $\boldsymbol{V}^s$
    end for
    if $\max_s \log H^s < \log(\epsilon_{\max})$ **and** $\max_s \log H^s > \log(\epsilon_{\min})$ **then**
      $\widetilde{G}(\boldsymbol{\theta}) = \frac{\sum_s \exp(\log H^s)}{I_{MCI}}$
    else
      $\widetilde{G}(\boldsymbol{\theta}) = \mathrm{e}^{\max_s \log H^s}$
    end if
    $G(\boldsymbol{\theta})^{(t)} = \frac{\widetilde{G}(\boldsymbol{\theta})}{t+1} + \frac{G(\boldsymbol{\theta})^{(t-1)}}{t+1} t$
    if $\frac{\left|G(\boldsymbol{\theta})^{(t)} - G(\boldsymbol{\theta})^{(t-1)}\right|}{G(\boldsymbol{\theta})^{(t-1)}} < \rho$ **then**
      return $G(\boldsymbol{\theta})^{(t)}$
    end if
  end for
  return $G(\boldsymbol{\theta})^{(t)}$

point range exceptions that affect the original sampling method. When the normalizing constant $G(\boldsymbol{\theta})$ is very large or very small, we have found that it is often impossible to calculate its estimate $\overline{H}$ without incurring floating-point range exceptions. A common approach in the queueing network literature to address floating point issues is to use scaling of demands, as first proposed by [Lam. 1982]. Unfortunately, this is always effective for single-class models, but not always effective in multiclass models. This is because no scaling factor is known to renormalize correctly all the classes and ensure that the normalizing constant falls within the floating point range on all instances. In this case, instead of approximating $G(\boldsymbol{\theta}) \approx \overline{H}$, we use $\log G(\boldsymbol{\theta}) \approx \log \max_s H^s$, since the maximum of the I_{MCI} samples H^s is representative of the order of magnitude of $G(\boldsymbol{\theta})$. Furthermore, our MCI implementation includes a stop condition that splits the I_{MCI} samples into several separate runs and checks after each run if the current estimate has converged within a tolerance ρ. To be more specific, if at the tth run the generated normalizing constant is $\widetilde{G}(\boldsymbol{\theta})$, then the cumulative moving average will be

$$G(\boldsymbol{\theta})^{(t)} = \frac{\widetilde{G}(\boldsymbol{\theta})}{t+1} + \frac{G(\boldsymbol{\theta})^{(t-1)}}{t+1} t.$$

In the next section, we refer to Algorithm 1 as IMCI. The main difference between IMCI and MCI is the choice of U_i, which is later in Section 3.4 shown to create a large performance difference between the two implementations in favour of IMCI.The floating-point range limit is set to $\epsilon_{\max} = \epsilon_{\min}^{-1} = 1.7 \cdot 10^{308}$. The number of samples generated for each iteration is denoted by I_{iter}. In the experiments reported later, we always set this parameter to be $I_{iter} = 500$. One thing to notice is that even for the same tolerance, IMCI returns randomized results for different runs. As we show later in Section 5 that this randomization significantly affects the Gibbs sampling process.

3.3. Taylor expansion method (TE)

In addition to the IMCI, we introduce a new approach to approximate the normalizing constant for computing the likelihood function (1). Our approach consists in applying a Taylor expansion of $G(\boldsymbol{\theta})$ along a given dimension of the demands $\boldsymbol{\theta}$, and iteratively use an AMVA algorithm to approximate the first-order derivative in the expansion. To the best of our knowledge, this is the first time that a technique is proposed to approximate normalizing constants without considering asymptotic limits or sampling.

The proposed approximation is denoted by TE and defined as follows. Recall from sensitivity analysis of queueing networks that [de Sousa e Silva and Muntz 1988]

$$\frac{dG(\boldsymbol{\theta})}{d\boldsymbol{\theta}_{ij}} = \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}} G(\boldsymbol{\theta}) \tag{7}$$

where $Q_{ij}(\boldsymbol{\theta})$ is the mean queue-length of jobs of class j at queueing station i. Consider now a first-order Taylor expansion of $G(\boldsymbol{\theta})$

$$G(\boldsymbol{\theta} + d\boldsymbol{\theta}_{ij}) = G(\boldsymbol{\theta}) + \frac{dG(\boldsymbol{\theta})}{d\boldsymbol{\theta}_{ij}} d\boldsymbol{\theta}_{ij} + o(\theta_{ij}^2)$$

where $d\boldsymbol{\theta}_{ij}$ updates only dimension ij. Using (7) and ignoring higher-order terms we obtain the approximation

$$G(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_{ij}) \approx G(\boldsymbol{\theta}) \left(1 + \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}}\right) \Delta\theta \tag{8}$$

where $\Delta\boldsymbol{\theta}_{ij} = \Delta\theta \cdot \mathbf{1}_{ij}$, in which $\mathbf{1}_{ij}$ is a vector of zeros except for a one in position ij. The quantity $\Delta\theta$ is referred to as the step size of the TE algorithm. The above expression readily implies that

$$\log G(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_{ij}) \approx \log G(\boldsymbol{\theta}) + \log\left(1 + \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}}\right) + \log \Delta\theta. \tag{9}$$

which can be iteratively evaluated at each step for increasing values of θ_{ij} if we are able to efficiently compute $Q_{ij}(\boldsymbol{\theta})$. This can be done using an AMVA algorithm, e.g., the Bard-Schweitzer or AQL [Cremonesi et al. 2002], which can return an estimate in fractions of seconds, with a computational complexity of just $O(MR)$ and $O(MR^2)$ per iteration, respectively. In our experiments, these AMVA algorithms typically completed in tens or hundreds of milliseconds even on large models with tens of queues and classes.

Based on this observation, it is possible to see that the value of $G(\boldsymbol{\theta})$ can be iteratively approximated as follows. We start from any value of $\boldsymbol{\theta}$ for which $G(\boldsymbol{\theta})$ is known, for example setting all θ_{ij} equal to zero such that

$$G(\mathbf{0}) = \frac{\prod_{j=1}^{R} Z_j^{N_j}}{N_j!}, \qquad \mathbf{0} = (0, \ldots, 0). \tag{10}$$

Then, we iterate (8) along all of the dimensions of $\boldsymbol{\theta}$, one at a time. For each dimension, we increase θ_{ij} by $\Delta\theta$ and compute $G(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_{ij})$ from the previously computed value $G(\boldsymbol{\theta})$ using (9), where the $Q_{i,j}(\boldsymbol{\theta})$ terms are computed by an AMVA algorithm, initialized with the mean queue-lengths found at the last invocation of AMVA. At the first invocation of AMVA, we initialize the algorithm with a balanced distribution of jobs across the stations. Finally, the value of $\log G(\boldsymbol{\theta})$ is obtained and either be used directly or exponentiated to obtain $G(\boldsymbol{\theta})$. As we show later, this iterative structure combines well with the Gibbs sampling algorithm. A pseudocode summarizing the TE approximation scheme is given in Algorithm 2. To the best of our knowledge, TE is the

Algorithm 2 Taylor expansion method

Require: $\Delta\theta$, $\boldsymbol{\theta}$, $\boldsymbol{Z}$, $\boldsymbol{K}$
  compute $\log G(\mathbf{0})$ by (10)
  $\boldsymbol{\theta}^* = \mathbf{0}$
  for $i = 1 : M$ **do**
    for $j = 1 : R$ **do**
      for $k = 1 : \theta_{ij}/\Delta\theta$ **do**
        compute $\log G(\boldsymbol{\theta^*} + \Delta\boldsymbol{\theta}_{ij})$ by (9)
        $\boldsymbol{\theta}^* = \boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}_{ij}$
      end for
    end for
  end for
  return $\log G(\boldsymbol{\theta})$

only existing method that can work directly with logarithms of normalizing constants and thus avoiding the floating-point range exceptions.

The above approach introduces linear execution time with the step size $\Delta\boldsymbol{\theta}_{ij}$, which may pose unacceptable overhead if $G(\boldsymbol{\theta})$ needs to be frequently evaluated. Noticing that (7) represents a standard ordinary differential equation (ODE), the execution time of evaluating $G(\boldsymbol{\theta})$ can be dramatically reduced by using existing ODE solvers such as MATLAB's *ode23*, which automatically determines the step size on behalf of the user. Other ODE solvers may also be used. Further, $\log(G(\boldsymbol{\theta}))$ can also be evaluated by the ODE solver, where the corresponding equation is obtained by taking the logarithm on both sides of (7), which generates

$$\frac{d\log(G(\boldsymbol{\theta}))}{d\boldsymbol{\theta}_{ij}} = \frac{Q_{ij}(\boldsymbol{\theta})}{\theta_{ij}}. \tag{11}$$

Solving for $\log(G(\boldsymbol{\theta}))$ in the ODE solver improves the numerical properties of the TE method by avoiding floating-point range exceptions. Therefore, from now on we will report results for TE using this ODE-based implementation based on logarithm.

3.4. Validation

To illustrate and compare the accuracy of IMCI, MCI and TE in approximating the normalizing constant, we consider both the computation of a *single* normalizing constant $G(\boldsymbol{\theta})$ and of a *sequence* of normalizing constants that differ for small changes in a demand value. The latter scenario is more representative of the use of the method in techniques like Gibbs sampling. For comparison, we consider a large number of randomly generated queueing networks.

Computation of a single normalizing constant.. We first consider the evaluation of a single normalizing constant. The randomly generated models have parameters as in Table I. The step size $\Delta\theta_{ij}$ used for TE is set to $\Delta\theta_{ij} = \rho\theta_{ij}$. For each model generated with the above parameters, 10 sub-models are defined by randomly generating the number of jobs and the demands from the uniform distribution. Without loss of generality, demands are normalized for each class j so that $\sum_{j=1}^{R}\theta_{ij} = 1$. Using this approach, we generate a total of 5400 models. The exact value of the normalizing constant is computed in each step by the convolution algorithm. During the experiment, we have tested TE with both the Bard-Schweitzer and AQL approximations for the queue length computations. We have found that AQL is significantly slower than Bard-Schweitzer and the overall accuracy of TE with the two methods is rather similar.

Table I: Parameters of generated queueing models

Parameter	*Description*	*Value*
$K = \sum_j K_j$	total number of jobs	$\{4, 20, 40\}$
M	number of servers	$\{2, 4, 8, 16, 32\}$
R	number of job class	$\{2, 3, 4\}$
Z_j	think time	$\{1, 10, 100, 1000\}$
ρ	tolerance	$\{0.1, 0.01, 0.001\}$

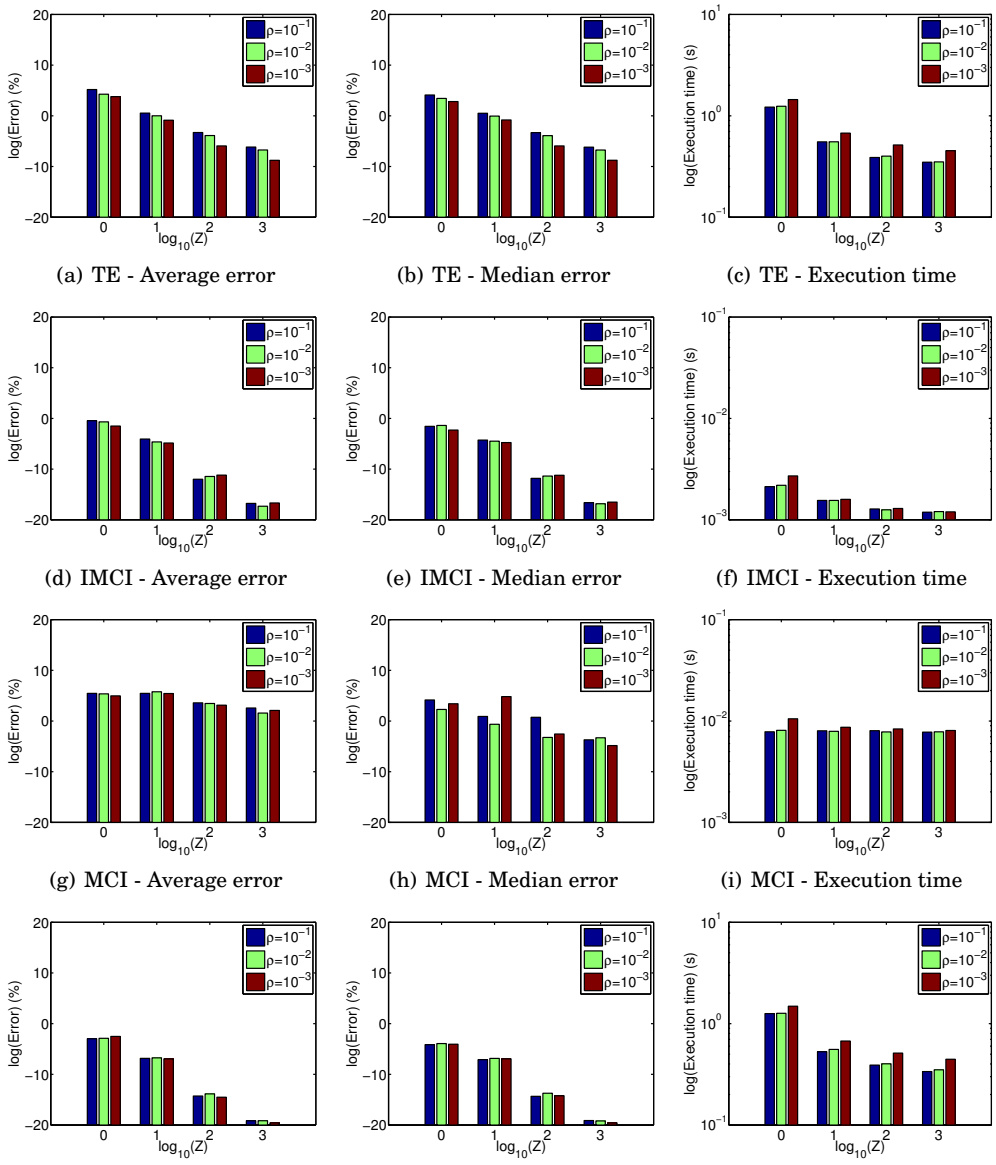
Therefore from now on we assume that TE is instantiated using the Bard-Schweitzer AMVA approximation (BS).

Figure 1 presents the average and median absolute errors of the normalizing constant estimates and the computational times for the IMCI, MCI and TE methods; memory occupation is negligible for all algorithms. It can be seen that our implementation of IMCI has generally the best accuracy and it is the fastest among the three algorithms. The accuracy of TE and MCI is similar, however for TE the accuracy gradually increases as the tolerance ρ becomes finer, while MCI shows fluctuations, which highlights its sensitivity to the initial warmup. With $\rho = 10^{-3}$, the TE method achieves a good accuracy of 10% mean error for all cases. If we restrict our attention to cases where $Z \neq 1$, in which the system has high CPU utilization, the error further decreases to 0.14%. On the other hand, the tolerance ρ has a smaller impact on IMCI, which enjoys similar accuracy and computation times for different ρ values. Another point to be noticed is that IMCI converges even for large values of ρ. Therefore, changing ρ does not significantly affect the accuracy of the algorithm. In terms of execution time, the decrease of the tolerance ρ leads to the rise of execution time for all the methods. We also show in Figure 1 the IMCI method with the same execution time as TE. Clearly given more time, IMCI is able to obtain a more accurate result than TE. Summarizing, the TE and MCI method show overall good accuracy, but they are dominated by IMCI for the approximation of a single normalizing constant. Considering that IMCI outperforms MCI, we shall consider only this implementation in the rest of paper.

Computation of a sequence of normalizing constants.. We now compare IMCI and TE in the scenario where one needs to compute a sequence of normalizing constants, a problem that arises in the application of Gibbs sampling to the demand estimation problem. This is because Gibbs sampling needs to calculate at each iteration a certain cumulative distribution function by solving an integral involving the normalizing constant, as we discuss in Section 4. To compare the performance of IMCI and TE in this scenario, we consider the similar problem of computing the integral of $\log G(\boldsymbol{\theta})$ across dimension θ_{ij}, discretizing the integration range with step size $\Delta\theta_{ij}$. Except for θ_{ij}, all the other demands are kept constant throughout the integration, such that

$$\int_{\theta_{ij}^-}^{\theta_{ij}^+} \log G(\theta_{ij}) \mathrm{d}\theta_{ij} \approx \sum_{k=1}^{\lceil (\theta_{ij}^+ - \theta_{ij}^-)/\Delta\theta_{ij} \rceil} \log G(\theta_{ij}^- + k\Delta\theta_{ij}) \Delta\theta_{ij}, \tag{12}$$

where we explicit the dependence of the normalizing constants only on θ_{ij}. The smaller $\Delta\theta_{ij}$ is, the more the integral will be accurate and we set it to be $\Delta\theta_{ij} = \rho\theta_{ij}$. This integral is used to illustrate the computation of a sequence of normalizing constants on models that are different for small changes of the θ_{ij} parameter. Figure 2 shows the average and median error and the computational time for both methods. The experiments are similar to the ones for computing single normalizing constants. However we



(a) TE - Average error (b) TE - Median error (c) TE - Execution time

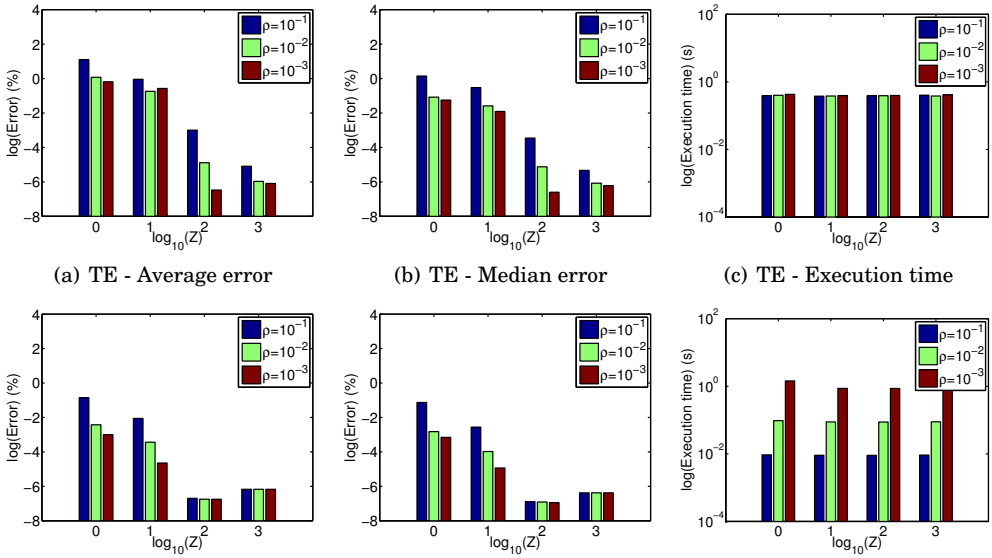
(d) IMCI - Average error (e) IMCI - Median error (f) IMCI - Execution time

(g) MCI - Average error (h) MCI - Median error (i) MCI - Execution time

(j) IMCI (same time as TE) - Average error (k) IMCI (same time as TE) - Median error (l) IMCI (same time as TE) - Execution time

Fig. 1: Computation of $G(\boldsymbol{\theta})$ - ρ is a convergence tolerance

do not consider $R = 4$ since it requires excessively large computational requirements to keep. The results in the figure is similar as Figure 1 for the accuracy. However, it can be noticed that the difference of the execution time between IMCI and TE is significantly reduced compared to Figure 1 and the execution time for TE is linear in ρ. This is due to the fact that IMCI needs to compute the normalizing constant for each iteration, since the samples generated at the previous step of the iteration cannot be



(a) TE - Average error (b) TE - Median error (c) TE - Execution time

(d) IMCI - Average error (e) IMCI - Median error (f) IMCI - Execution time

Fig. 2: Computation of integrals - ρ is a convergence tolerance

Table II: Summary of main notation concerning Gibbs sampling

S	number of samples to be generated
D	number of observations provided to the algorithm
$\boldsymbol{N}$	input data set
$\Delta\theta$	step size
$\boldsymbol{I}$	discretization range

reused, while TE uses ODE solvers to efficiently update the value of the normalizing constant by (8) with a single execution of AMVA.

4. DEMAND ESTIMATION WITH GIBBS SAMPLING

Leveraging the TE and IMCI methods, we are now ready to introduce a demand estimation method based on Gibbs sampling [Geman and Geman 1984]. That is, we investigate the applicability of Gibbs sampling for inference of the service demand matrix $\boldsymbol{\theta}$ from observation of a sequence $\boldsymbol{N} = \{\boldsymbol{n}^{(1)}, \boldsymbol{n}^{(2)}, \ldots, \boldsymbol{n}^{(D)}\}$ of independent states for the queueing network. We denote with $\boldsymbol{n}^{(i)}$ the i-th observed state, $1 \leq i \leq D$. A summary of the main notation used throughout this section is given in Table II.

4.1. General algorithm

The application of Gibbs sampling to the problem at hand is as follows. Let S be the number of samples to be drawn by Gibbs sampling and call θ_{ij}^s the sth sample generated upon estimating the demand θ_{ij}. In Gibbs sampling, each sample θ_{ij}^s is distributed as $P(\theta_{ij}|\boldsymbol{\theta}_{(ij)}^s, \boldsymbol{N})$, where $\boldsymbol{\theta}_{(ij)}^s = (\theta_{11}^s, \ldots, \theta_{i,j-1}^s, \theta_{i,j+1}^{s-1}, \ldots, \theta_{MR}^{s-1})$ depends on the past generated samples. The samples generated by the Gibbs method will have an empirical distribution that will converge to the posterior density function for the demands θ_{ij}, given the priors and the observations [Asmussen et al. 2007]. The main difficulty of applying Gibbs sampling to demand estimation is to efficiently compute the condi-

tional distributions $P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$. We show below that this can be efficiently done by the TE method. To obtain $P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ for each demand θ_{ij}, we condition as follows

$$P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) = \frac{P(\theta_{ij}, \boldsymbol{\theta}^s_{(ij)}|\boldsymbol{N})}{\int P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)}|\boldsymbol{N})\mathrm{d}\tilde{\theta}_{ij}} \tag{13}$$

where the integral is on the range of the allowed values of θ_{ij}, which may be application dependent. Then using Bayes theorem

$$\begin{aligned} P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) &= \frac{P(\theta_{ij}, \boldsymbol{\theta}^s_{(ij)}|\boldsymbol{N})}{\int P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)}|\boldsymbol{N})\mathrm{d}\tilde{\theta}_{ij}} = \frac{\frac{P(\boldsymbol{N}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})P(\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})}{P(\boldsymbol{N})}}{\int \frac{P(\boldsymbol{N}|\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})}{P(\boldsymbol{N})}\mathrm{d}\tilde{\theta}_{ij}} \\ &= \frac{P(\boldsymbol{N}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})P(\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})}{\int P(\boldsymbol{N}|\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})\mathrm{d}\tilde{\theta}_{ij}}. \end{aligned} \tag{14}$$

Notice that the integral may be written as a constant independent of θ_{ij}, thus we have

$$P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) \propto P(\boldsymbol{N}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})P(\theta_{ij}, \boldsymbol{\theta}^s_{(ij)}). \tag{15}$$

Let us now observe that, due to the assumed independence of the observations

$$P(\boldsymbol{N}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)}) = \prod_{\boldsymbol{n}\in\boldsymbol{N}} P(\boldsymbol{n}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)}), \tag{16}$$

where $P(\boldsymbol{n}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})$ is the likelihood of observing state $\boldsymbol{n}$ for a given assignment of θ_{ij} and given the last samples drawn for the other demands. Thus, we have

$$P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) \propto \prod_{\boldsymbol{n}\in\boldsymbol{N}} P(\boldsymbol{n}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})P(\theta_{ij}, \boldsymbol{\theta}^s_{(ij)}) \tag{17}$$

where the likelihood function $P(\boldsymbol{n}|\theta_{ij}, \boldsymbol{\theta}^s_{(ij)})$ appearing in (17) can be readily computed from the equilibrium distribution of the network given in (1).

The final step is to draw a sample from $P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$. Since θ_{ij} is a finite quantity, being dimensionally a time, our methodology assumes it to lie in a finite interval $\boldsymbol{I} = [\theta^-_{ij}, \theta^+_{ij}]$. If no information is available on this range upon performing the estimation, one can chose a large enough interval to express a high uncertainty. Conditioning on $\theta_{ij} \in \boldsymbol{I}$, we can define the cumulative distribution of the demands in this interval as

$$F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) = \int_{\theta^-_{ij}}^{\theta_{ij}} P(\tilde{\theta}_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})\mathrm{d}\tilde{\theta}_{ij}. \tag{18}$$

We combine this expression with (17), instantiated with the expression of the likelihood (1). This brings

$$F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) = \frac{1}{H(\boldsymbol{\theta}^s_{(ij)})} \int_{\theta^-_{ij}}^{\theta_{ij}} \prod_{\boldsymbol{n}\in\boldsymbol{N}} \left(G^{-1}(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})\tilde{\theta}^{n_{ij}}_{ij}\right) P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})\mathrm{d}\tilde{\theta}_{ij} \tag{19}$$

where

$$H(\boldsymbol{\theta}^s_{(ij)}) = \int_{\theta^-_{ij}}^{\theta^+_{ij}} \prod_{\boldsymbol{n}\in\boldsymbol{N}} \left(G^{-1}(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})\tilde{\theta}^{n_{ij}}_{ij}\right) P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})\mathrm{d}\tilde{\theta}_{ij}$$

is a normalizing constant ensuring that $F(\theta^+_{ij}) = 1$. In the last expression, we make explicit in the argument of the normalizing constant the dependence on the random variable $\tilde{\theta}_{ij}$ and on the past samples $\boldsymbol{\theta}^s_{(ij)}$. Further, it is important to note that the last two expressions do not include the contribution of the factorials in (1) and of the

demands different from $\tilde{\theta}_{ij}$. This is because these terms can be simplified by grouping them in front of both the integral in (19) and the integral that defines $H(\boldsymbol{\theta}^s_{(ij)})$. Therefore, by breaking the product of all the observations in $\boldsymbol{N}$ we see that the cumulative distribution expression may be rewritten as

$$F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) = \frac{1}{H(\boldsymbol{\theta}^s_{(ij)})} \int_{\theta^-_{ij}}^{\theta_{ij}} (G^{-1}(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)}))^D \prod_{\boldsymbol{n}\in\boldsymbol{N}} \tilde{\theta}_{ij}^{n_{ij}} P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)}) \mathrm{d}\tilde{\theta}_{ij} \tag{20}$$

$$= \frac{1}{H(\boldsymbol{\theta}^s_{(ij)})} \int_{\theta^-_{ij}}^{\theta_{ij}} (G^{-1}(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)}) \tilde{\theta}_{ij}^{\overline{n}_{ij}})^D P(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)}) \mathrm{d}\tilde{\theta}_{ij}, \tag{21}$$

which depends only on the observed mean queue-lengths $\overline{n}_{ij} = \sum_{\boldsymbol{n}\in\boldsymbol{N}} n_{ij}/D$ and the number of observations D. Since the dataset appears in the above expression only via the mean queue-lengths $\overline{n}_{ij}$, this implies that the application of Gibbs sampling to demand estimation only requires measurement of mean queue-lengths, not of the individual states $\boldsymbol{n}$ for the system under consideration. This is an important conclusion of our analysis, which makes the technique very simple to apply to real systems, since one needs to measure mean queue-lengths, rather than detailed system states.

Algorithm 3 presents the pseudo code for the Gibbs sampling algorithm. At the end of the algorithm, a set of samples θ^s_{ij} is returned. These can be used to numerically estimate the posterior density of the demands, for example to apply the maximum a posteriori method, or they can be simply averaged to determine the demand estimate directly. To account for the initial burn-in period due to the transient of the underlying Markov chain, in our experiments we discard the first half of all the generated samples. Thus, for an experiment that generates $S = 50$ samples for each demand, only the last 25 will be used to estimate the service demands.

Note that a prior distribution $P(\boldsymbol{\theta})$ is required for the application of the Gibbs sampling approach described in this section. Priors can be important especially when the empirical dataset $\boldsymbol{N}$ is small, since known information about the distribution of parameters can help in better discriminating among the feasible estimates. In this paper, we primarily focus on the case where the user chooses a uniform distribution for the prior, thus each parameter θ_{ij} is equilikely in a range $\boldsymbol{I} = [\theta^-_{ij}, \theta^+_{ij}]$. This corresponds to a maximum likelihood estimator in $\boldsymbol{I}$. However, a strength of our methodology is that it can accept arbitrary prior distributions. We illustrate the benefits of this feature for Dirichlet priors in Section 5.3.5.

4.1.1. Implementation with TE. Each integral appearing in (20) can be evaluated by computing the integrand at a number of equispaced points, chosen with step $\Delta\theta_{ij}$, in the integral range $\boldsymbol{I}$. To minimize the risk of floating-point range exceptions, we first compute the logarithm of the integrand and then exponentiate the resulting value, summing its contribution to the integral. Since the normalizing constant $H(\boldsymbol{\theta}^s_{(ij)})$ is required to evaluate the cumulative distribution at any point, we evaluate $H(\boldsymbol{\theta}^s_{(ij)})$ first and record during the calculation the partial sums of the integral. These partial sums allow us to evaluate the integral in the definition of $F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ with a small effort.

Considering the two approaches for approximating $G(\boldsymbol{\theta})$ introduced in Section 3, TE is able to compute more efficiently than IMCI the integrals in $H(\boldsymbol{\theta}^s_{(ij)})$ and $F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ with small tolerance, since it can iteratively update the value of $\log G(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})$. Conversely, IMCI needs to draw a new set of samples for each $\tilde{\theta}_{ij}$. The method assumes that all the integration ranges are discretized, thus samples $\theta^s_{i,j}$ are approximated to the closest point in the discretized range. In each iteration of the inner loop of Algorithm 3, we run TE twice starting from the demand vector $(\theta^{s-1}_{ij}, \boldsymbol{\theta}^s_{(ij)})$

Algorithm 3 Gibbs Sampling

Require: $\boldsymbol{N}, \boldsymbol{I}, M, R, \boldsymbol{K}, S, \boldsymbol{Z}, \Delta\theta_{ij}$
  $\boldsymbol{\theta}^0 = (\theta^-_{11}, \ldots, \theta^-_{MR})$
  compute $\log G(\boldsymbol{\theta}^0)$
  for $s = 1 : S$ **do**
    for $i = 1 : M$ **do**
      for $j = 1 : R$ **do**
        $\boldsymbol{\theta}^s_{(ij)} = (\theta^s_{11}, \ldots, \theta^s_{i,j-1}, \theta^{s-1}_{i,j+1}, \ldots, \theta^{s-1}_{MR})$
        compute $\log G(\tilde{\theta}_{ij}, \boldsymbol{\theta}^s_{(ij)})$ for all $\tilde{\theta}_{ij} \in [\theta^-_{ij}, \theta^+_{ij}]$ with step $\Delta\theta_{ij}$
        generate a pseudo-random number $u \in [0, 1]$
        find $\theta^s_{ij} : F(\theta^s_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N}) \leq u < F(\theta^s_{ij} + \Delta\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$
      end for
    end for
  end for
  return $\theta^s_{ij}, \forall, s, i, j$

and computing with step $\Delta\theta_{ij}$ all the normalizing constants up to the demand vectors $(\theta^-_{ij}, \boldsymbol{\theta}^s_{(ij)})$ in the first run and up to $(\theta^+_{ij}, \boldsymbol{\theta}^s_{(ij)})$ in the second run. Using these two runs, we can utilize the value of $\log G(\tilde{\theta}^{s-1}_{ij}, \boldsymbol{\theta}^s_{(ij)})$ computed at the last iteration in order to initialize TE at the following run. The normalizing constants computed with these two runs of TE are exactly the ones needed to determine $F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ for any θ_{ij}. We can then proceed to sample θ^s_{ij} and move to the next iteration, discarding all the normalizing constants computed in the two runs except $\log G(\theta^s_{ij}, \boldsymbol{\theta}^s_{(ij)})$ that will be used to initialize TE at the next iteration.

Finally, we remark that in our experiments often the cumulative distribution $F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ converges to 1 for $\theta_{ij} \ll \theta^+_{ij}$. This leads to the situation where a significant amount of effort is spent to compute an integral over a range of demand values that have a very small tail probability. To address this problem, we assume that the user provides a tolerance parameter τ and we compute $F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ up to $\min(2\theta^{\tau,s-1}_{ij}, \theta^+_{ij})$, where $\theta^{\tau,s-1}_{ij}$ is the quantile for the probability mass $1 - \tau$ obtained at the last iteration of the algorithm. For instance, $\tau = 10^{-10}$ means that the upper bound will be between the minimum of the specified θ^+_{ij} and 2 times the $1 - 10^{-10}$ quantile of $F(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ for the next iteration.

5. EXPERIMENTAL EVALUATION

5.1. Methodology

We have evaluated the performance of Gibbs sampling for service demand estimation using random experiments, described in this section, and a case study, discussed in Section 6. Our experiments have been run on a desktop machine with an Intel Core i7-2600 CPU, running at 3.4 GHz and with 16 GB of memory. We have tested single and multiple classes cases using simulated data. To establish performance across typical scenarios, we have considered topologies where all nodes are in series or all nodes are in parallel, which are both quite common in models of real applications.

5.1.1. Evaluation criteria. We use the mean absolute percentage error as the evaluation metric for the accuracy of the estimators, which is

$$\epsilon = \frac{1}{MR} \sum_{i=1}^{M} \sum_{j=1}^{R} \frac{|\theta_{ij} - \theta^*_{ij}|}{\theta_{ij}} \tag{22}$$

where θ_{ij} is the exact service demand value and θ^*_{ij} is the value estimated by the algorithm. We always discard the first half of the demand samples to avoid the bias of the initial burn-in and calculate the mean demand on the second half of the samples.

5.2. Terms of comparison

For comparison, we have also implemented other MCMC algorithms [Asmussen et al. 2007; Brooks et al. 2011] and applied them to demand estimation, namely Metropolis-Hastings sampling [Hastings. 2012; Metropolis et al. 1953] and Slice sampling [Neal 2003; Damlen et al. 1999]. The computation of the normalizing constant is also required by these two approaches and will be approximated with TE and IMCI.

5.2.1. Metropolis Hastings (MH). The MH algorithm constructs a Markov chain with continuous state space in order to sample from a target distribution $f(\boldsymbol{\theta})$. In the case of demand estimation, this target distribution is $P(\boldsymbol{\theta}|\boldsymbol{N})$, which from Bayes theorem

$$P(\boldsymbol{\theta}|\boldsymbol{N}) = \frac{P(\boldsymbol{N}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{N})} = \frac{\prod_{\boldsymbol{n}\in\boldsymbol{N}} P(\boldsymbol{n}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{N})} \tag{23}$$

where $P(\boldsymbol{N})$ is a normalizing constant to ensure $\sum_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\boldsymbol{N}) = 1$. A random walk is realized as follows. At iteration $t+1$, the algorithm randomly generates a new parameter $\boldsymbol{\theta}$ from a distribution $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ proposed by the user, where $\boldsymbol{\theta}^{(t)}$ is the demand vector sampled at the previous iteration. The algorithm next computes an acceptance rate

$$\alpha = \min\left(1, \frac{P(\boldsymbol{\theta}|\boldsymbol{N})q(\boldsymbol{\theta}^{(t)}|\boldsymbol{\theta})}{P(\boldsymbol{\theta}^{(t)}|\boldsymbol{N})q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})}\right) \tag{24}$$

The algorithm accepts the candidate $\boldsymbol{\theta}$ with probability α and sets $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}$ if the sample is accepted, or otherwise rejects it by setting $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)}$. After S iterations the algorithm has produced an output sample of size S of vectors $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(S)}$ distributed according to the target distribution $f(\boldsymbol{\theta})$.

Compared to Gibbs sampling, the MH algorithm changes several demand values at each iteration. This does not allow us to update the value of the normalizing constant with TE in an efficient manner. In fact, we show later that this method performs best when implemented with IMCI. In the implementation for demand estimation in our experiment, the proposed distribution for the MH sampling is set to be a multivariate Gaussian distribution with a covariance matrix $\beta\boldsymbol{I}$, where where $\boldsymbol{I}$ is the identity matrix. In default, we set $\beta = 10^{-3}$ to match the magnitude of the demands.

5.2.2. Slice sampling. Slice sampling is one of the MCMC methods where a distribution is sampled uniformly from the region under the plot of its density function. The advantage of Slice sampling is that it does not need to discretize the integration interval. With uniform sampling from the area under the distribution Slice sampling is able to automatically match the characteristics of the distribution.

For the target distribution $P(\boldsymbol{\theta}|\boldsymbol{N})$ as shown in (23), the procedure of slice sampling is depicted in Figure 3 and can be summarized as follows:

(1) Choose a random start point $\boldsymbol{\theta}^0$
(2) Choose h uniformly between 0 and $P(\boldsymbol{\theta}^0|\boldsymbol{N})$
(3) Draw a horizontal line across h
(4) Sample a point from the line segments

In practice, it is difficult to sample directly from a slice since the line segments are difficult to be located. Therefore we use the stepping-out and shrinking-in strategy proposed by Neal in [Neal 2003]. This requires an additional parameter to define the

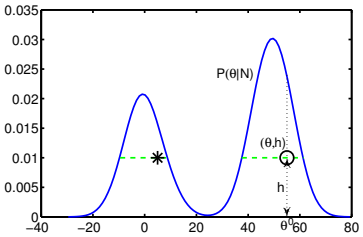


Fig. 3: Slice sampling procedure

width of the slice. Considering that a coarse width makes the sampling process inaccurate while a small one leads to slow convergence, we choose the width to be of the same magnitude as the demands. Therefore in our validation presented later, we set the width to be 0.1.

We compare Slice sampling with Gibbs sampling because with TE the normalizing constant integration could be approximated very efficiently. However, Slice sampling has the advantage that this integration is not required, which theoretically should make Slice sampling more efficient, although we did not find it to be, as we show later.

5.3. Numerical Validation

We now present the evaluation of the three sampling algorithms on the simulated queue length data for demand estimation. The data is generated from the underlying Markov Chain of a closed network with R classes and M queueing stations along with one delay node. The simulation method is standard [Bolch et al. 2006]. In total, we simulate $500,000$ service completions. The parameters considered here are described in Table III. The think time is assumed to be $Z_j = 1$ for each class of jobs. For each model generated from the above parameters, 10 sub-models are defined by randomly generating the number of jobs and the demands from the uniform distribution. Without loss of generality, demands are normalized for each class j so that $\sum_{j=1}^{R} \theta_{ij} = 1$. To obtain queue-length samples, we have first computed the steady state probability from the simulation events. Then we have sampled from it by generating random numbers between 0 and 1 and determining which sample fits in the cumulative probability. This is also known as the inverse transform sampling. We generate 5000 queue length samples in the following experiments.

For fair comparison, a time limit of 15 minutes is set. The tolerance of TE and IMCI is set to $\rho = 10^{-3}$. For the Gibbs method, the discretization range $\boldsymbol{I}$ always has a lower bound $\theta_{ij}^{-} = 0$ and the initial upper bound is $\theta_{ij}^{+} = 1$ for all the dimensions ij. We set $\tau = 1 - 10^{-10}$ for the probability tolerance. The initial point for the three algorithms is set to $\mathbf{0}.1$ for all the classes.

We show the convergence plots of the three algorithms for $M = 4, R = 4, K = 40, C_i = 4$ case in Figure 4. Since the total demand has 16 dimensions, we only plot the demand of the first queue and the first class. From the figure we can see that Gibbs sampling is able to converge quickly and experiences a small oscillation around the exact value. The MH algorithm cannot reach the exact value and gets stuck at a local optimum. Slice sampling is able to achieve a better performance than MH sampling and gradually converges to the exact value, but with a slower rate than Gibbs.

5.3.1. Multicore approximation. So far we have described the Gibbs method for estimating demands of queueing stations with a single server, which are reasonable abstractions of single core machines. However, servers are now prevalently multi-core, which raises the need to incorporate information about the number of cores in the model.

Table III: Parameters of queueing models for Gibbs sampling validation

Parameter	*Description*	*Value*
$K = \sum_j K_j$	total number of jobs	$\{4, 20, 40\}$
M	number of servers	$\{2, 4, 8\}$
R	number of job class	$\{2, 3, 4\}$
C_i	number of cores	$\{1, 2, 4\}$

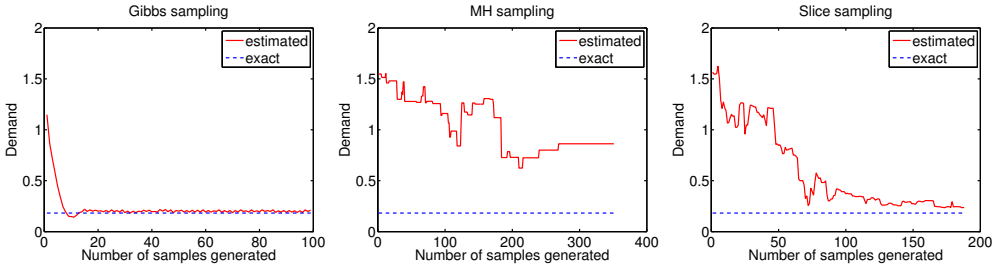


Fig. 4: Convergence plots

Since the scheduling of queueing stations is processor sharing, the error incurred by scaling the demand by the number of cores tends to be small. However, in the context of demand estimation, the problem is that a limited jobs in execution at a station would be classified with a demand that is much smaller than the true value. To address this problem, we scale the demands by the average number of cores used by the application based on the collected dataset, rather than by the total number of cores. This approach takes better into account the actual load of the system provided by the observations.

Let us assume first that we have measured a dataset $\boldsymbol{N} = \{\boldsymbol{n}^{(1)}, \boldsymbol{n}^{(2)}, \ldots, \boldsymbol{n}^{(D)}\}$ of queue-length states and assume that each state $\boldsymbol{n}^{(i)}$ is associated to a timestamp t_i for the instant of collection. Then we can readily estimate from this data the probability $\pi_i(n_i)$ that n_i jobs are in node i. The value n_i is a total number of jobs, irrespectively of their class. Then, given the total number of servers (i.e., cores) c_i at station i, we can estimate $\bar{c}_i$, the number of cores used on average by the application, as

$$\bar{c}_i = (1 - \pi_i(0))^{-1} \sum_{s=1}^{K} \pi_i(s) \min(c_i, s) \tag{25}$$

where $(1 - \pi_i(0))^{-1}$ is the probability that the server is not idle and the summation computes the average number of cores used given that the server is busy. Using $\bar{c}_i$, the demand for the multicore instances may be estimated as $\theta_{ij} = \bar{c}_i \theta_{ij}^{(1)}$, where $\theta_{ij}^{(1)}$ is estimated by Gibbs sampling on a model that assumes all stations to have a single server. In case only measurements of mean queue-lengths $\bar{n}_{ij}$ are available, as opposed to the whole dataset $\boldsymbol{N}$, one could approximate the above estimate as $\bar{c}_i = \min(c_i, \bar{n}_i)$, where $\bar{n}_i = \sum_{j=1}^{R} \bar{n}_{ij}$ is the total mean queue-length at station i. If also the average utilization U_i is measured, then this estimate can be refined as $\bar{c}_i = \min(c_i, U_i^{-1} \bar{n}_i)$.

Finally, we remark that we also considered the case where each core is modelled as a single-server queue. However, the demand matrix will contain $\sum_{i=1}^{M} c_i R$ dimensions in this case, which increased the computation time significantly in our initial experiments and limited the ability of the Gibbs method to find good estimates for the demands in a timescale of minutes. We have therefore not investigated this approach any further.

5.3.2. Evaluation Results. We here run the Gibbs, MH and Slice sampling algorithms on random models and present the results, classified according to the utilization U_{max} of the bottleneck station in the model. We chose the maximum utilization as the criteria for model classification since we have found empirically that the algorithms are rather sensitive to this parameter. The three levels of load we considered are low ($U_{max} \leq 50\%$), medium ($50\% < U_{max} \leq 75\%$), and high ($U_{max} > 75\%$). In all experiments, the prior distribution is assumed to be a uniform prior inside the discretization range $\boldsymbol{I}$.

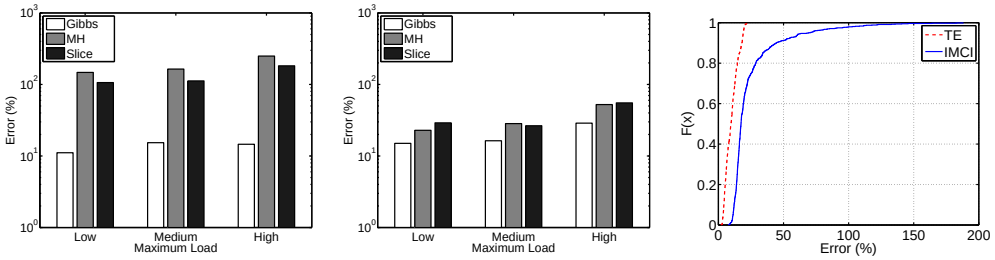
Figure 5 reports the average error for different bottleneck utilizations with $\rho = 10^{-3}$. The experiments reveal that Gibbs sampling implemented with the TE approximation is able to achieve approximately a 11% error for different $U_{\max}$levels. This is the most accurate method among all the ones we tested. It is better than Gibbs with IMCI, which has around 20% error, despite that IMCI shows higher accuracy in Section 3. To understand this result, we examine the coefficient of variation (CV) of the generated samples for the Gibbs method with TE and IMCI in Figure 6(a). As the figure suggests that the samples generated from IMCI have a large variance, we interpret this to be caused by the randomness of the normalizing constant generated by IMCI. To validate this claim, we consider a case with $M = 4, R = 4, C = 1, K = 40$. For each iteration of Gibbs sampling, we recompute 10 times $P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$ in (17) by rerunning IMCI every time. At each time, we compute the expectation of this distribution, i.e. $\mathbb{E}(\theta_{ij}) = \theta_{ij} P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$, which leads to 10 expectations. We then compute the span of these values by $\max(\mathbb{E}(\theta_{ij})) - \min(\mathbb{E}(\theta_{ij}))$. Finally we divide the span by the sampling interval $(\theta^+_{ij} - \theta^-_{ij})$ to show the intrinsic variability of the estimate. Figure 6(b) and 6(c) report the relative span of θ_{11} and θ_{22} as the Gibbs sampling process evolves. It can be seen that the relative span of TE is zero due to its deterministic nature. However, the random sampling process of IMCI contributes large variations of $P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$, which produces samples in a wide range, a fact that misguides the search and eventually leads to worse demand estimates in IMCI than in TE. Moreover, the variation does not reduce as the sampling process converges.

For MH and Slice sampling, the IMCI approximation gives a much lower error than TE since it is more accurate and efficient in computing the single normalizing constant needed by these methods. Further, for MH and Slice sampling TE degrades the performance under high loads. This cannot be attributed to TE itself, since this degradation is not visible for Gibbs with TE, therefore we conclude that this due to the fact that these methods generally perform worse than Gibbs sampling in heavy load, as also visible when implemented with IMCI. The fact that TE is slower than IMCI for MH and Slice sampling can explain the accuracy degradation.

To further substantiate the claim that TE is superior to IMCI for use within Gibbs sampling, Figure 5(c) presents the cumulative distribution across all the models of the error of our Gibbs sampling implementations based on TE and IMCI. It can be readily seen that the implementation of Gibbs sampling with TE is superior also in error distribution to the implementation based on IMCI. We have also done the same experiments with $\rho = 10^{-2}$ and the result is qualitatively similar as above.

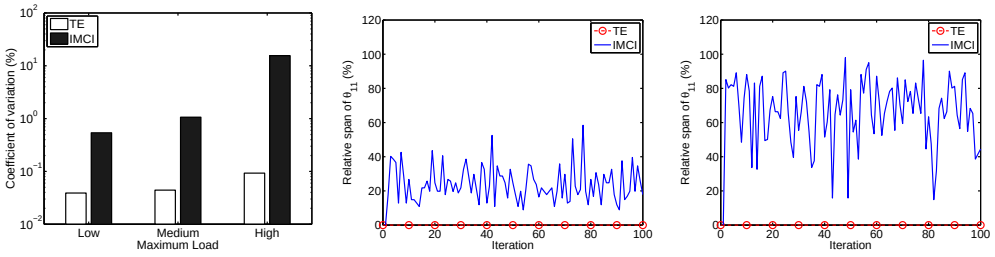
Summarizing, the experiments reveal that the Gibbs sampler is superior to MH and Slice sampling in demand estimation tasks. Furthermore, the proposed TE approximation delivers a visible decrease of error compared to an approximation based on an existing method, IMCI. Our experiments suggest that the inherent variability of the IMCI method significantly affects the accuracy of the demand estimates.

5.3.3. Sensitivity analysis on large models. To understand the performance of the proposed methods on large models, we expand the evaluation by considering models with $M \in \{16, 32\}, K \in \{80, 160\}$,$R = 4$ with 10 sub models. Figure 7 presents the evalu-



(a) TE (b) IMCI (c) CDF of Gibbs Sampling

Fig. 5: Evaluation of the algorithms on different models



(a) CV (b) $P(\theta_{ij}|\boldsymbol{\theta}^s_{(11)}, \boldsymbol{N})$ (c) $P(\theta_{ij}|\boldsymbol{\theta}^s_{(22)}, \boldsymbol{N})$

Fig. 6: Evaluation of TE and IMCI with Gibbs sampling

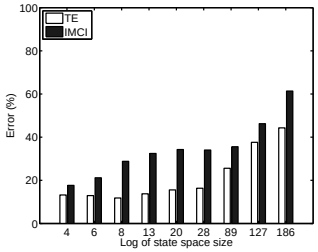


Fig. 7: Sensitivity analysis on large models

ation results, where we quantify the model size with the size of state space, which is computed as $\prod_{r=1}^{R} \binom{N_r+M}{M}$, in which we count the presence of the infinite server that models Z. We keep the execution time fixed with 15 minutes. It can be seen that as the model size increases the error gradually arises as well. Still, on models of practical relevance it is clear that TE is much better than IMCI. The practical limit is around the model with $M = 16$ queues, $R = 4$ classes and $K = 160$ jobs where TE exceeds 30%. However, for the same threshold IMCI exceeds the error in a much smaller model. One thing to notice is that, since we keep the number of simulated events fixed in the experiment ($500,000$ service completions), as the models size increases the generated queue length is less accurate for representing the exponentially increases state space.

5.3.4. Random initial points. Since all the estimation algorithms require an initial point as input, we explore the influence on the results of choosing a different initial point. In particular, we have randomized the initial point around θ_{ij} with a uniform distribution which has $\theta_{ij}/10$ as lower bound and $10\theta_{ij}$ as higher bound on each class j, where θ_{ij}

is the exact demand. The experiment assumes $M = 4, R = 3, K = 20, C_i = \{1, 2, 4\}, \rho = 10^{-3}$. For each model generated from the above parameters, 10 sub-models are defined. For each sub-model, we initialize it with 10 different random initial points. Table IV reports the average error and the standard deviation for seven random initial points for each experiment. From the table it is possible to observe that the standard deviation for Gibbs sampling are relatively small compared to the mean error, which means that the initial points do not appear to have a major impact on the results. The high standard deviation of the MH and Slice sampling with TE method indicates that they are still strongly influenced by the initial value of the chain and are not mixing properly.

Table IV: Randomized initial point

		1 core			2 cores			4 cores		
		Gibbs	MH	Slice	Gibbs	MH	Slice	Gibbs	MH	Slice
TE-BS	*Average(%)*	4.0	375	250	11.5	641	448	19.7	887	713
$\rho = 10^{-3}$	*STD(%)*	0.5	81.0	80.6	2.3	135	151	3.6	241	207
IMCI	*Average(%)*	31.3	56.4	74.3	18.3	24.8	45.0	20.6	23.4	23.7
$\rho = 10^{-3}$	*STD(%)*	4.7	8.9	53.0	2.0	1.9	77.6	4.3	4.7	6.8

According to [Gelman et al. 2014], MH sampling should be tuned so that the acceptance rate is in the range of 25%-50%. To study the impact of the acceptance rate on the accuracy, we varied the β parameter that defines the covariance matrix $\beta \boldsymbol{I}$ of the proposed multivariate normal distribution. The smaller β is, the higher the acceptance rate will be. Table V reports the result for different models with increased acceptance rate compared to the default $\beta = 10^{-3}$ case. We only show the case of IMCI, since TE takes a long time to compute a single normalizing constant. The difference of β in Table V is to make sure that the acceptance rate fall in the range suggested by [Gelman et al. 2014]. From the result it can be noticed that with the suggested acceptance rate, the MH sampling is still not as good as Gibbs.

Table V: Impact of acceptance rate of MH sampling on accuracy

		1 core	2 cores	4 cores
IMCI $\rho = 10^{-2}$	*Average(%)*	78.2	29.2	43.1
	STD(%)	8.7	3.5	4.1
	Acceptance rate(%)	29.7	35.6	31.7
	β	10^{-3}	10^{-3}	10^{-3}
IMCI $\rho = 10^{-3}$	*Average(%)*	53.3	23.6	25.7
	STD(%)	10.3	2.5	5.3
	Acceptance rate(%)	37.3	28.0	33.4
	β	10^{-4}	10^{-4}	10^{-5}

5.3.5. Sensitivity analysis of prior distribution. Using a prior distribution $P(\boldsymbol{\theta})$ is particularly useful when the number of dataset $\boldsymbol{N}$ is small. To show the actual effectiveness of the prior distribution and analyse how sensitive the inference of the posterior mean is to changes in the prior distribution, we use a Dirichlet distribution as prior. The Dirichlet distribution is the conjugate prior of the multinomial distribution and therefore appears to be a natural choice given that the BCMP product form is a scaled product of multinomial terms. The probability density function of the Dirichlet distribution of order n for variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ with parameters $\boldsymbol{q} = (q_1, \ldots, q_n)$ is defined as

$$f(\boldsymbol{x}; \boldsymbol{q}) = \frac{1}{B(\boldsymbol{q})} \prod_{i=1}^{n} x_i^{q_i - 1} \tag{26}$$

where $B(\boldsymbol{q})$ is a normalizing constant

$$B(\boldsymbol{q}) = \frac{\prod_{i=1}^{n} \Gamma(q_i)}{\Gamma(\sum_{i=1}^{n} q_i)} \tag{27}$$

and $\Gamma(x)$ denotes the Gamma function with constraint that $\sum_{i=1}^{n} x_n = 1$. The mean value and the variance for each variable x_i of the Dirichlet distribution is $E[x_i] = q_i/q_0$, where $q_0 = \sum_i^n q_i$. To estimate the parameters of the Dirichlet distribution, we use the method proposed in [Minka 2000] to estimate $\boldsymbol{q}$. In real applications, the demands data used to fit $\boldsymbol{q}$ could come from historical values of the estimates. For the sake of illustration, we here simulate the demand data from a run of Gibbs sampling with a uniform prior. This data is then normalised so that $\sum_{i=1}^{M} \sum_{j=1}^{R} \theta_{ij} = 1$ and we fit this dataset to a Dirichlet distribution. By scaling the think times Z_j by the same constant, the estimation can be restricted on the interval $[0, 1]$ for each demand.

We consider the following experiments. Let θ_{ij} be the true demand and $\bar{\theta}_{ij}$ be the mean value of the prior distribution for the same demand. We define the deviation

$$\eta = \frac{\sum_{i=1}^{M} \sum_{j=1}^{R} \left|\bar{\theta}_{ij} - \theta_{ij}\right|}{MR},$$

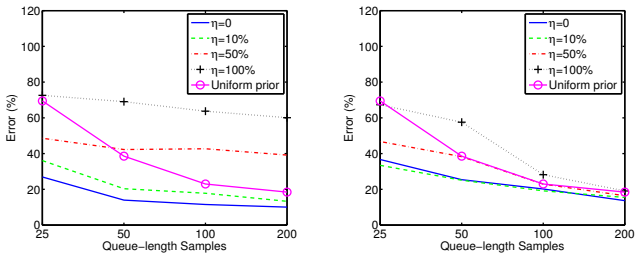
as a measure of the distance of the prior to the true value of the demand. Using the dataset generated by Gibbs sampling, which has typically low variance in the distribution of the samples, we fit the Dirichlet parameter vector and denote it by $\boldsymbol{q}_{low}$. Note that low variance represents low uncertainty on the location of the demands. We also consider a modified vector $\boldsymbol{q}_{high} = 100\boldsymbol{q}_{low}$ that increases the variance, while keeping the same mean value for all the demands. This modified vector allows us to evaluate the accuracy of the Gibbs method in presence of high uncertainty in the prior.

We then run experiments for the Gibbs method instantiated with the TE method and run for 15 minutes. The algorithm is tested on closed queueing networks specified by the following combination of parameters: $R \in \{2, 3\}, M \in \{2, 4\}, C_i \in \{1, 2\}, K \in \{4, 20\}$. We assume all the stations have the same number of cores C_i. The think time is assumed to be $Z_j = 1$ for each class. The queue-length data is generated from the Markov Chain simulation. The number of queue-length samples is set to be $D \in \{25, 50, 100, 200\}$. The deviation η is chosen in $\{10\%, 50\%, 100\%\}$.

Experimental results are illustrated in Figure 8, which also includes the uniform prior case. From both figures, it is clear that a prior distribution with small deviation η improves significantly the accuracy of the estimate compared to the uniform prior case, but as expected only if the dataset is small ($D \leq 50$). This is because the dependence of the prior is progressively lost as new samples are generated. The experiments also reveal that a large η, if coupled with small variance in the prior ($\boldsymbol{q}_{low}$ case), will bias the conditional probability $P(\theta_{ij}|\boldsymbol{\theta}^s_{(ij)}, \boldsymbol{N})$. This will obviously result in a poor estimate. However, a large variance ($\boldsymbol{q}_{high}$ case) will help to reduce this bias and when the number of observed data increases, the sampling process will tend to converge to the true values. Overall, the experiments in this section prove that just a few tens of samples are sufficient to obtain reasonably accurate demand estimates. This may be relevant in particular for real-time decision-making, where only a few measurement samples may be obtained before taking a tuning decision for a computer system.

6. CASE STUDY

In this section, we evaluate the Gibbs method against data collected from a real application, Apache Open for Business (OFBiz) [Apache OFBiz 2014], which is an open source enterprise resource planning system. We generated a set of user requests for this application using the OFBench tool [Moschetta and Casale 2012] with an instal-



(a) $\boldsymbol{q}_{high} = \boldsymbol{q}_{low}$ (b) $\boldsymbol{q}_{high} = 100\boldsymbol{q}_{low}$

Fig. 8: Dirichlet Prior distribution evaluation - all estimates converge unless the exact demand has very low probability prior

Table VI: Case study experiment results

		2 cores (c1.medium)			8 cores (c1.xlarge)		
		A. Gibbs	MH	Slice	A. Gibbs	MH	Slice
TE $\rho = 10^{-3}$	*Time(m)*	5	5	5	5	5	5
	Error(%)	14.0	18.5	21.5	14.9	38.0	248
	Time(m)	15	15	15	15	15	15
	Error(%)	14.1	17.0	14.5	14.8	34.6	162
IMCI $\rho = 10^{-3}$	*Time(m)*	5	5	5	5	5	5
	Error(%)	15.0	15.9	17.3	15.1	18.5	16.9
	Time(m)	15	15	15	15	15	15
	Error(%)	14.9	15.6	17.1	15.0	17.2	16.5

lation where the web server is co-located with the default Apache Derby database. The experiment was run on Amazon EC2, with OFBiz running simultaneously on two load-balanced virtual machines, a *c1.medium* instance and a *c1.xlarge* instance.

Modelling the inter-submission time of requests as a think time, the whole system may be seen as a queueing network model, with a fixed population K given by the number of clients of the OFBench tool. Each class represents a request type and per-class populations are obtained by multiplying K with the probability that a certain type of request is issued. Therefore, K_j represents the average number of requests of a given type in the system as a whole.

We use OFBench to send 8 classes of requests to OFBiz and we parse the OFBiz logs to get the $\boldsymbol{N}$ dataset. The true value θ_{ij} of the service demand is estimated using the Complete Information (CI) algorithm proposed in [Perez et al. 2013], which is able to return the near exact demand from the dataset given the full sample path of the system from the application logs. In order to avoid assuming knowledge on the population of the model, which may be unrealistic in some applications, the model population K is estimated from the empirical dataset as the maximum number of concurrently executing requests in the system across all the recordings. To get the think time, the throughput $\boldsymbol{T}$ is also obtained from the logfiles, so that by Little's Law $Z_j = n_{0j}/T_j$ where $n_{0j}, 1 \leq j \leq R$ represents the average number of users in thinking state.

We then evaluate the Gibbs, MH and Slice sampling algorithms against the true values estimated by CI. The parameters for the algorithms are the same as the experiment for simulated data. Results are shown in Table VI. From the table we see that the Gibbs sampling implementation based on TE produces the best results. This is consistent with the numerical validation results. It can also be noticed that MH and Slice sampling enjoy an increase of accuracy compared to the numerical results. We interpret this improvement due to the the fact that here in case study only one queue-

ing station is considered, i.e. $M = 1$. With such a simple model, MH and Slice is able to have a fast convergence to the demand estimate.

7. RELATED WORK

Existing approaches for demand estimation may be categorized in the following groups, according to the estimation approach used.

Regression-based estimation. In [Rolia and Vetland 1995], linear regression is introduced for resource demand estimation and papers such as [Rolia and Vetland 1998] and [Pacifici et al. 2008] propose to use multivariate linear regression to estimate mean service demands. In [Zhang et al. 2007], a regression based methodology is presented to estimate the CPU demand in multi-tier software systems. Experiments with the TPC-W benchmark demonstrate the robustness of such approximation under diverse workloads with changing transaction mixes. Moreover, the work in [Casale et al. 2008] uses Least Trimmed Squares regression based on CPU utilization and system throughput to estimate service times. The study in [Pacifici et al. 2008] considers dynamically estimating CPU demands by formulating the problem as a multivariate linear regression of CPU utilization against throughput. However, these methods are known to suffer from the multicollinearity problem leading to biased estimates [Kalbasi et al. 2011]. Compared to regression methods, our technique requires only queue-length information and can be more efficient with few samples thanks to the ability of incorporating prior distributions in the analysis.

Machine learning. The work in [Sharma et al. 2008] proposes to use independent component analysis to infer workload characteristics based on CPU utilization and the number of customers. The study in [Cremonesi et al. 2010] proposes a method using density-based clustering and clusterwise regression to estimate demands. In [Sutton and Jordan 2011], the authors propose a method to use Bayesian inference for estimation of the demand in the system. By utilizing MCMC to sample from the posterior distribution, this method proves to be robust to missing data. [Cremonesi and Sansottera 2012] proposes an algorithm to estimate the service demands in presence of system updates and changes. A time-based linear clustering algorithm is employed to identify different linear clusters for each service demands. In [Ross et al. 2007], an estimation of arrival and service rate approach based on queue length is presented by defining a Ornstein-Uhlenbeck diffusion to achieve the approximation. Unlike our method, [Ross et al. 2007] is only for open queueing networks. Other literature with Bayesian inference on open models include the works of [Armero et al. 1994] and [Insua et al. 1998]. Compared to the above methods, our approach is the first one to apply MCMC to closed models with multi-class workloads.

Optimization methods. The work in [Liu et al. 2006] proposes an approach to use quadratic programming to estimate the service demands. End-to-end response time as well as utilization of the nodes and system throughput are required inputs. Both [Wu and Woodside 2008] and [Zheng et al. 2008] use an Extended Kalman Filter for parameter tracking. In [Wu and Woodside 2008], a calibration framework based on fixed test configurations is proposed. [Zheng et al. 2008] applies tracking filters on time-varying systems. [Zheng et al. 2008] extends [Zheng et al. 2005] and adapts the Kalman filter to estimate the performance model parameters based on CPU utilization and response time. In [Kalbasi et al. 2012], a novel iterative approach based on linear programming is proposed for multi-tier systems. An extensive evaluation demonstrates the effectiveness of the method. Compared to these works, the Gibbs algorithm allows us to include prior information on the parameters and with only queue-length data.

Finally, the work in [Spinner et al. 2015] presents a detailed exploration of the state-of-the-art in resource demand estimation technologies. The accuracy of the algorithms

is compared by analyzing them in the same environment and varying the parameters of the test environment to see the sensitivity of the methods estimates.

8. CONCLUSION

In this paper, we have presented a service demand estimation methodology for closed multi-class queueing networks based on Gibbs sampling. Differently from existing approaches, our method requires only queue length data, which is easy to collect in real systems. The challenge is to evaluate a likelihood function of the queue length data due to the difficulty of computing the normalizing constant in closed models. To tackle this problem, we have proposed a new normalizing constant approximation method based on Taylor expansion for use in conjunction with Gibbs sampling. Through numerical validation and a case study, we have shown the effectiveness of the proposed algorithm compared with other Markov Chain Monte Carlo methods.

Acknowledgement

The research leading to these results has received funding from the European Commission as part of the DICE action (H2020-644869).

REFERENCES

C. Armero, and MJ. Bayarri. 1994. Bayesian prediction in M/M/1 queues. *Queueing Systems* 15, 1 (1975), 401–417.

S. Asmussen, and P. W. Glynn. 2007. *Stochastic simulation: Algorithms and analysis*.

F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. 1975. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *JACM* 22, 2 (1975), 248–260.

G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. 2006. *Queueing Networks and Markov Chains*. 2nd ed., John Wiley and Sons.

S. Brooks, A. Gelman, G. Jones, X. Meng. 2011. *Handbook of Markov Chain Monte Carlo*. CRC press.

G. Casale. 2006. An Efficient Algorithm for the Exact Analysis of Multiclass Queueing Networks with Large Population Sizes. In *Proc. of joint ACM SIGMETRICS/IFIP Performance*. ACM Press, 169–180.

G. Casale, P. Cremonesi, and R. Turrin. 2008. Robust Workload Estimation in Queueing Network Performance Models. In *Proc. of Euromicro PDP*. 183–187.

P. Cremonesi, K. Dhyani, and A. Sansottera. 2010. Service time estimation with a refinement enhanced hybrid clustering algorithm. *Proc. of ASMTA* (2010), 291–305.

P. Cremonesi and A. Sansottera. 2012. Indirect Estimation of Service Demands in the Presence of Structural Changes. In *Proc. of QEST*. IEEE, 249–259.

P. Cremonesi, P. J. Schweitzer, and G. Serazzi. 2002. A Unifying Framework for the Approximate Solution of Closed Multiclass Queuing Networks. *IEEE Trans. on Computers* 51 (2002), 1423–1434.

P. Damlen, J. C. Wakefield, and S. G. Walker. 1999. Gibbs sampling for Bayesian nonconjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society, B* 61 (1999), 331–344.

E. de Sousa e Silva and R. R. Muntz. 1988. Simple Relationships Among Moments of Queue Lengths in Product Form Queueing Networks. *IEEE Trans. on Computers* 37, 9 (1988), 1125–1129.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6 (1984), 721–741.

A. Gelman, J. Carlin, H. Stern, D. Rubin. 2014. *Bayesian data analysis*. Taylor & Francis.

W. Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57, 1 (1970), 97–109.

D. R. Insua, M. Wiper, and F. Ruggeri. 1998. Bayesian analysis of M/Er/1 and M/H_k/1 queues. *Queueing Systems* 30, 3 (1998), 289–308.

G. Koole, and A. Mandelbaum. 2012. Queueing models of call centers: An introduction. *Annals of Operations Research* 113, 1 (2002), 41–59.

A. Kalbasi, D. Krishnamurthy, J. Rolia, and S. Dawson. 2012. DEC: Service Demand Estimation with Confidence. *IEEE Trans. on Software Engineering* 38, 3 (2012), 561–578.

A. Kalbasi, D. Krishnamurthy, J. Rolia, and M. Richter. 2011. MODE: mix driven on-line resource demand estimation. In *Proc. of CNSM*. International Federation for Information Processing, 1–9.

C. Knessl and C. Tier. 1992. Asymptotic Expansions for Large Closed Queueing Networks with Multiple Job Classes. *IEEE Trans. on Computers* 41, 4 (1992), 480–488.

S. Kraft, S. Pacheco-Sanchez, G. Casale, and S. Dawson. 2009. Estimating service resource consumption from response time measurements. In *Proc. of Valuetools*. ACM, 48.

S. Lam. 1982. Dynamic scaling and growth behavior of queuing network normalization constants. *JACM* 29, 2 (1982), 492–513.

Y. Liu, I. Gorton, and A. Fekete. 2005. Design-level performance prediction of component-based applications. *IEEE Trans. on Software Engineering* 31, 11 (2005), 928–941.

Z. Liu, L. Wynter, C. Xia, and F. Zhang. 2006. Parameter inference of queueing models for IT systems using end-to-end measurements. *Performance Evaluation* 63, 1 (2006), 36–60.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, 6 (1953), 1087–1092.

T. Minka. 2000. Estimating a Dirichlet distribution. (2000).

J. Moschetta and G. Casale. 2012. OFBench: an Enterprise Application Benchmark for Cloud Resource Management Studies. In *Proc. of SYNASC*. IEEE, 393–399.

R. Neal. 2003. Slice sampling. *Annals of statistics* (2003), 705–741.

G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. 2008. Cpu demand for web serving: Measurement analysis and dynamic estimation. *Performance Evaluation* 65, 6 (2008), 531–553.

J. Perez, S. Pacheco-Sanchez, and G. Casale. 2013. An Offline Demand Estimation Method for Multi-threaded Applications. In *Proc. of MASCOTS*. IEEE Computer Society, Washington, DC, USA, 21–30.

Apache OFBiz project. 2014. http://ofbiz.apache.org.

J. Rolia and V. Vetland. 1995. Parameter estimation for performance models of distributed application systems. In *Proc. of CASCON*. IBM Press, 54.

J. Rolia and V. Vetland. 1998. Correlating resource demand information with ARM data for application services. In *Proc. of WOSP*. ACM, 219–230.

J. Rolia and K. Sevcik. 1995. The Method of Layers. *IEEE Trans. on Soft. Eng.* 21, 8 (Aug. 1995), 689–700.

J. Ross, T. Taimre, and P. Pollett. 2007. Estimation for queues from queue length data. *Queueing Systems* 55, 2 (2007), 131–138.

K. Ross, D. Tsang, and J. Wang. 1994. Monte Carlo summation and integration applied to multiclass queuing networks. *JACM* 41, 6 (1994), 1110–1135.

K. Ross and J. Wang. 1997. Implementation of Monte Carlo integration for the analysis of product-form queueing networks. *Performance Evaluation* 29, 4 (1997), 273–292.

A. Sharma, R. Bhagwan, M. Choudhury, L. Golubchik, R. Govindan, and G. Voelker. 2008. Automatic request categorization in internet services. *Performance Evaluation Review* 36, 2 (2008), 16–25.

S. Spinner, G. Casale, F. Brosig, and S. Kounev. 2015. Evaluating Approaches to Resource Demand Estimation. Performance Evaluation. *Performance Evaluation* 92, 10(2015), 51–71.

C. Sutton and M. Jordan. 2011. Bayesian inference for queueing networks and modeling of internet services. *The Annals of Applied Statistics* 5, 1 (2011), 254–282.

B. Tuffin. 1997. Variance reduction applied to product form multiclass queuing networks. *ACM Trans. on Modeling and Computer Simulation* 7, 4 (1997), 478–500.

B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi. 2005. An analytical model for multi-tier internet services and its applications. In *Proc. of ACM SIGMETRICS*. ACM Press, 291–302.

W. Wang, and G. Casale. 2013. Bayesian service demand estimation using gibbs sampling. In *Proc. of IEEE MASCOTS*. IEEE Press, 567–576.

X. Wu and M. Woodside. 2008. A calibration framework for capturing and calibrating software performance models. *Computer Performance Engineering* (2008), 32–47.

Q. Zhang, L. Cherkasova, and E. Smirni. 2007. A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications. In *Proc. of ICAC*. 27–27.

T. Zheng, C. Woodside, and M. Litoiu. 2008. Performance model estimation and tracking using optimal filters. *IEEE Trans. on Software Engineering* 34, 3 (2008), 391–406.

T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. 2005. Tracking time-varying parameters in software systems with extended Kalman filters. In *Proc. of CASCON*. IBM Press, 334–345.